AverStar, Inc.

# Software Interface Analysis Tool

## Installation Guide

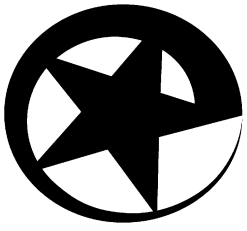
**Prepared under NASA Contract: NAS2-96024, Task Order 8**

**Document : NAS2-96024-99-192, 30 September 1999**

**Prepared for:**

**NASA Ames Research Center, Moffett Field, CA 94035-1000**

AverStar



# Table of Contents

# Installation Instructions

## Overview

The SIAT tools require a Netscape Enterprise Server, version 3.5 or higher on a Sun Solaris platform. Certain functions also require Perl. Perl is available at http://www.perl.com/latest.html.

The installation deals with files in several different areas. Before beginning the installation, you should have the following information for your system:

- Location of the SIAT Netscape Enterprise Server files. Usually, when a Netscape Enterprise server is setup, a directory is created with a name like `https-servername`. Under the server directory there are directories for configuration files, `config` directory, and log files, `log` directory. You will need access to all these areas.

NOTE: If there is already an HTTP server running on the server machine, the recommended configuration for SIAT is to create a new server running on a different port number that is dedicated to SIAT. This will also allow multiple versions of SIAT to be available, as they can be running on different HTTP servers, listening to different port numbers.

- Location of the document root for the SIAT Netscape Enterprise server. This is usually not in the same place as the server files since this represents the data to be served by the HTTP server.

- SIAT Home directory. If SIAT has been installed previously on the system, there will be a SIAT home directory. If not, this directory will be created as part of the installation.

- AdaMagic home directory. This location is needed when configuring the SIAT tool.

- New versions of SIAT that increase the major version number imply substantial changes to the tool. When the major version number changes, all source code installed in the system will need to be recompiled and re-analyzed for use with the new version of the system.

It is also useful, although not required, to have access to the Netscape Administration Server if one is running on your system. It provides a convenient way to look at the settings for the HTTP servers running on the system, and for stopping and starting those servers.

The three main pieces of the SIAT are:

- DSM: this is a Netscape Server Plugin. It is shipped as a shared library that is linked into the Netscape Enterprise server at run time. The installation instructions explain how to configure the server to use the DSM.

- XREF: this is the cross-reference engine. It is an executable that performs the cross-reference queries on the Ada library. The instructions explain how to configure the DSM to use the XREF.

- Source files: these are HTML, Javascript, and Ada source files that are used by the SIAT. These files all belong under the HTTP server's document root directory so they can be served out to users.

## Getting Started

1) Go to or create the SIAT home directory on the server machine. In that directory, extract the files from the tape (or tar file) included with these instructions. You should have the following subdirectories:

   - applets Contains applets associated with SIAT tools
   - utils Contains HTML and Javascript files used by the tools
   - sources Ada source files for predefined libraries (e.g., RTS) and example software components
   - bin scripts used by the cross-reference engine, and utilities used for building the source code base for analysis
   - dsm Dynamic Service Module server plug-in files
   - xref The cross-reference engine files
   - docs Electronic versions of the Installation Guide and User's Guide

2) Under the document root of the HTTP server create a directory `siat`.

3) Move the applets, utils and sources directories under the siat directory created in step 2. Or, if this is not a new installation, replace the contents of the applets and utils directories with the contents of the corresponding directories extracted from the tape. If there is a new RTS directory under the sources directory extracted from the tape, move it to the corresponding place under the sources subdirectory.

> NOTE: if a new version of the Ada compiler was delivered with an updated version, the code under the sources directory will probably have to be recompiled. Refer to the release notes or readme that came with the delivery.

4) If this is a first installation, you can create a temporary `index.html` file in the siat directory by linking to the file `top_index.html` in the utils directory:

```
ln –s utils/top_index.html ./index.html
```

You will probably want to tailor your own index.html for your site. Use the temporary one as an example of the links to include to access the SIAT system. If this is not a first installation you probably have a tailored index.html that you should continue to use.

5) In the document root directory create an `error.html` file by linking to the file `error.html` in the utils directory:

```
ln –s siat/utils/error.html error.html
```

6) Under the siat directory, create a subdirectory called `users`. Make sure this directory is writeable by the user that runs the web server (often called nobody). This is where user directories will be created for users of the SIAT system. If this is not a new installation you can just keep your current users directory.

7) The siat tools will sometimes create temporary files that will need to be served by the HTTP server. The temporary files are created in the default place for temporary files on your system, such as `/var/tmp`. Create a link from the HTTP server document root to the temporary file directory so that it has the same name relative to the document root. For example, if the temporary files are created in `/var/tmp`, in the document root do the following:

```
mkdir var
cd var
ln –s /var/tmp tmp
```

8) Enable the following of symbolic links in the HTTP server. In the Netscape Administration Server, select the web server to get its configuration settings. Near the bottom of the frame on the left is a link called *Symbolic Links*, click on this to see the symbolic link settings. Both soft and hard links should be enabled.

## Install the Cross-reference Engine

The xref subdirectory of the SIAT home directory contains the files described below. They do not have to be in a specially named place since full pathnames are given in the later configuration parameters, but they must not be under the siat directory that is served by the HTTP server.

xref — The cross-reference engine executable. There is one xref engine running for each user on the system, it uses libbrowser.so.

xref.cfg — The xref engine config file. It contains information about what CSCIs are available for browsing and analysis. This file is modified each time a new

set of sources is made available on the system. Instructions for modifying the xref.cfg file are given below. Either keep this file along with the xref executable, or put it in the HTTP server's config directory.

libbrowser.so The shared library that implements all the browser query functions. It is shared by all copies of the xref engine that are running. Install this library where you would normally install shared libraries, or keep it with the xref executable.

## SIAT bin directory

This directory contains scripts that are used by the cross-reference engine for creating certain reports, and scripts to help the system administrator in building the source code tree for analysis. Several of the scripts in this directory use the perl language. If perl is not installed in /usr/local/bin/perl on your system, then change the first line of each of the perl scripts to reflect the proper path for the perl executable. The following scripts are in this directory:

grep_engine A perl script that implements the string search function across Ada library files.

format_gid A perl script to create the external data items report from the raw data contained in the GXREF.MAP file.

mk_GXREF.perl A perl script that takes the memory map files (provided with NCS and CCS), and converts the data into the canonical format required by the GXREF.MAP file. This version of the script only works on the memory map format that has been provided with NCS and CCS. Other developers memory maps will probably have a different format, and will probably require a different translation script.

mkindex A perl script that creates the index for a CSCI library after all the units have been compiled.

compile.csh A template for a compilation c-shell script to create and compile a CSCI library. Copy it to the appropriate CSCI source directory, and edit it as specified in the script itself.

## Install the Dynamic Service Module

The Dynamic Service Module (DSM) is a server plug-in for the Netscape Enterprise server. We are assuming a standard server installation where the server runs as user *nobody*, the server name is *https-siat* and the document root is */usr/local/netscape/docs*. Since the DSM saves state information between HTTP requests, and multiple HTTP server processes do not share memory, the server **must** be running as a single process.

1) M3ake a `lib` subdirectory of the https-siat server directory, and copy `libdsm.so` from the dsm directory into the lib directory. Ensure the permissions are set to allow user *nobody* to read the file.

2) Copy `error_msgs` from the dsm directory to the lib directory as well.

3) Modify the `obj.conf` file in the server's config directory to add the following two lines at the end of the group of other init lines at the top of the file. Make the appropriate substitutions in the two lines as described below. The bin directory contains a file `obj_conf.dsm` that contains these two lines for editing and copying to the obj.conf file for the server.

```
Init fn=load-modules  shlib=../lib/libdsm.so  funcs="dsm,dsm_init"
 Init fn=dsm_init \
    db_name=users@usersi \
    db_user=dbaccess \
    db_auth=siat98 \
    max_states=100 \
    document_root="/usr/local/netscape/docs”\
    error_msg_file="../lib/error_msgs" \
    xref_engine="/path/to/xref" \
    xref_config="/path/to/xref.cfg" \
    xref_library_path="/path/to/dynamic/libraries" \
    xref_timeout=28800 \
    ada_magic="/ADA_MAGIC/environment/variable" \
    utils_path="/path/to/xref_scripts/" \
    debug="on"
```

Note that the lines can be broken into multiple physical lines for readability by splitting them with a backslash and newline at the end of each physical line.

The first line specifies the path to the libdsm.so file. If you installed the libdsm.so file in the lib subdirectory as describe above the relative path in the example is correct since the server runs in the config directory. The **funcs** parameter specifies the names of the entry points in the libdsm.so file, and these should be exactly as specified above.

The second line specifies the parameters that are passed to the dsm_init function. These parameters are:

db_params — These parameters are place holders for when the users are managed in a database. They are currently expected, but the values are ignored.

max_states — This is an internal DSM parameter and should be left as is.

document_root — The document root directory for this server, */usr/local/netscape/docs* in our example.

error_msg_file — The path to the file containing the error messages in tag/ message pairs. Blank lines and comment lines (beginning with #) can separate the tag/message pairs. Each error message is identified by a *tag* which is the index into the error message file. Each tag is immediately followed by the text of the error message itself. The tags must **not** be changed, but the messages themselves can be changed as desired. The message text can be only one line, but it can be very long. If there is no message text, the tag is used instead. An initial version of the

error_msgs file was copied into the lib directory from the dsm directory above.

xref_engine — The full pathname for the xref engine executable, `xref`.

xref_config — The full pathname for the xref engine configuration file, `xref.cfg`.

xref_library_path — The full pathname for the directory in which the `libbrowser.so` shared library was installed.

xref_timeout — The length of time in seconds that an xref process should wait for input before killing itself. This parameter is optional. Its default value is zero which implies no timeout. If the timeout is not set and the user simply closes the web-browser without running the exit command, then the xref process becomes orphaned, and it will not go away until the HTTP server is restarted. This means that if the users don't use the exit command, more and more processes are used by the server. It is probably a good idea to set a timeout for the xref process. The example above is an eight hour timeout. That means the process will wait eight hours after the last user input before shutting down. The timeout should probably be pretty lengthy because you want users to be able to leave, such as for a meeting, and be able to continue their session a few hours later.

ada_magic — The value of the $ADA_MAGIC environment variable on this system. It should be the path to find the AdaMagic tools.

utils_path — The path to the bin directory (from the tape) that contains the cross-reference scripts. Note that the path should end with '/'.

debug — If the debug parameter is found, very verbose debugging output will be written in the server's error log. The value of the debug parameter is irrelevant, its presence turns debug logging on. To turn it off, remove the parameter.

4) Continue to modify the obj.conf file by adding the following service line to the default object group in the group of other service lines. This line should precede any `CM_*` service lines, and follow the two lines with `fn=imagemap` and `index-common`.

```
Service fn=dsm method="(GET|HEAD|POST)" type="text/*"
```

5) Add any necessary MIME types to the `mime.types` file, either with the admin server, or by directly editing the file. The additions should include any special file formats in use. Currently there are no additional MIME types needed by the SIAT system.

6) The DSM expects the SIAT to be available, so do not start up the server without having installed the cross-reference engine, and the source files under the document root.

7) Shutdown and restart the server to activate the changes. To minimize the disruption to users during a restart you can use the following steps. These steps ensure that there is always a libdsm.so under a running server, and it keeps around an old libdsm.so in case there is a need to revert.

- make any changes to obj.conf
- change directory to the lib directory
- `ln libdsm.so libdsm.so.old`
- cp <new libdsm from tape> new
- mv new libdsm.so
- ../stop; ../start

### Setting up a directory outside the DSM

If the HTTP server has been set up to be dedicated to SIAT, then the following steps are unnecessary. All traffic to the SIAT server is SIAT related and therefore would need to go through the DSM, all other traffic will be handled by other servers. If SIAT is running on an HTTP server that is serving HTML files for other uses, then implementing the following configuration may improve response on non-SIAT files.

The DSM has been set up to respond to all GET, POST and HEAD requests that come to the server. It also opens each file served out to process any variable substitutions in the file. In general, this is harmless, even if there are files in directories under the document root that are not part of the SIAT system. If there are files that you do not want the DSM to open before serving, you can set up the directory they are in to be served outside the DSM. For example, if there was a subdirectory `/no-dsm` in the document root, add the following directive to the obj.conf file to have it ignored by the DSM:

```
<Object ppath="/usr/local/netscape/docs/no-dsm/*">
  Service fn="send-file"
</Object>
```

The DSM will not open any files under the `no-dsm` directory. Obviously, the “no-dsm” directory can have any name you want, and there can be several of them. They all need to have separate object defined for them in the obj.conf configuration file.

## Adding Software Components to the system

The `sources` subdirectory in the siat area of the HTTP server should contain the sources for the various Computer Software Configuration Items (CSCIs) available for browsing and analysis. The files can be put in the sources directory, or links to the directories

where the sources reside on the system can also be made as long as the structure needed by the tools is maintained. The structure of the sources area is as follows:

- Immediately under the sources directory are directories for each uniquely named CSCI.
- Within each CSCI directory are subdirectories named for each version of the system available on the server.
- The version subdirectory contains all the ada sources and the Ada library for the CSCI. Generated files (e.g., HTML listings, xref files etc.) will be put in appropriate subdirectories.

## Building a new CSCI

The file `compile.csh` in the siat bin directory is a script template to compile a CSCI to create the Ada library and related files that are needed by the system. Copy the script file to the CSCI/version subdirectory where the Ada library will be built and follow these instructions to customize the script.

1) If the $ADA_MAGIC environment variable is not already set in your user envirnoment, uncomment the line that sets the variable, and set it to the correct value for AdaMagic on your system.

2) Set the name of the log file that records the output of running this script.

3) The script creates an ADA.LIB file explicitly. Modify the pathname for the Run Time System to be the full system pathname of the RTS installed under the SIAT sources directory. In this example that path would be:

```
/usr/local/netscape/docs/siat/sources/rts/int
```

4) Copy the modified Alsys sources into the current directory (if applicable – Space Station Only).

5) Set the sources variable to the list of source files to be analyzed. Wildcards can be used.

6) If the sources came from a non-Unix system, then they should be converted to Unix files. For example, the extra carriage return in DOS files should be removed before the files are compiled. Sun Unix systems have a utility `dos2unix` that can be used to convert the files:

```
dos2unix <original-file>  <converted-file>
Note: if the converted file has the same name as the original, the original file is rewritten after
conversion.
```

7) Run the script. Check that HTML files were created in the HTML subdirectory and that they have HTML contents. Also check the output for Ada compiler errors and warnings during compilation.

8) Run the script `mkindex` from the siat bin directory to create an index.html that corresponds to the contents of the library. Mkindex is a perl script, if perl is not installed as /usr/local/bin/perl on the system edit the first line to reflect the proper place to find the perl executable. Mkindex takes four parameters, the name of the CSCI, the version, the absolute system path to the directory that contains the library, and the UNIT.MAP file to be read to generate the library listing. The output should be directed to `HTML/index.html`. For example, to run mkindex for a CSCI named demo in the directory containing the Ada library: Run mkindex in the directory above the HTML directory, and where the ADA.LIB and UNIT.MAP files are to be found

```
mkindex Demo 1.0 /usr/local/netscape/docs/siat/sources/demo/1.0 UNIT.MAP > HTML/index.html
```

9) (Optional) If there are memory map reports that have been generated by the Lerose tool, they can be used to create a GXREF.MAP file for the library to map between Ada engineering name and PUI. The script mk_GXREF.perl is a perl script. Correct the first line of the script if perl is not installed in /usr/local/bin/perl. mk_GXREF.perl should be run in the same directory as the ADA.LIB file. It takes a list of memory map reports, and creates GXREF.MAP in the current directory. Any map files that do not follow the format expected will generate an error and will be ignored.

```
mk_GXREF.perl reports/*.MAP
```

At this point, if there are no errors, the CSCI has been built for analysis by the system. To make this CSCI version available to users the configuration information needs to be added to the `xref.cfg` file.

## Modifying xref.cfg

The xref.cfg file is used by the Xref engine provide the following information about each CSCI available on the system:

- Name
- Version
- The absolute system path of the library
- The URL path to the HTML files for constructing HTML links

The configuration file is made up of multiple CSCI configuration blocks. Each block has the form:

```
CSCI: <CSCI_Name>
  <configuration lines>
END_CSCI
```

<CSCI_Name> is a string identifying the CSCI.

Within each block is one or more configuration lines, of the form:

<Path-to-Ada-library> <Version-number> <URL-Path-to-HTML-files>

Path-to-Ada-library is the absolute path to the directory where the ADA.LIB file is located for the CSCI. This is the path as it would be given on the unix host system.

Version-number is a floating point number indicating the version of the CSCI

URL-Path-to-HTML-files is the URL path to the directory containing the HTML files relative to the server's document root.

Lines of text outside of CSCI blocks are ignored, allowing comments outside of CSCI blocks. Additional text inside CSCI blocks will cause errors.

An example configuration file might look like:

```
#Xref Demonstration

CSCI: Demo
/usr/local/netscape/docs/siat/sources/demo/1.0 1.0 /siat/sources/demo/1.0/HTML
/usr/local/netscape/docs/siat/sources/demo/2.0 2.0 /siat/sources/demo/2.0/HTML
/usr/local/netscape/docs/siat/sources/demo/.3.5 3.5 /siat/sources/demo/3.5/HTML
END_CSCI
```